Elektron User Manual

Elektron Codebase Version 1.4.0

Table of Contents

Introduction

Audience

The Elektron Users guide is intended to give experienced Java developers an introduction to the Elektron Integrated Development Environment. This guide assumes that the reader has a detailed knowledge of programming in Java 1.1 or later.

The Elektron Users Guide is to be read in conjunction with the Amber Development Guide and the Amber API Documentation; however, a basic understanding of Amber would be useful prior to reading this guide in order to get the most out of it. Where appropriate, this manual provides references to the relevant sections of these other documents.

Overview of Elektron

Elektron is a basic Integrated Development Environment (IDE) to assist Amber developers in producing Amber applications more easily than they could otherwise.

Elektron is itself an Amber application, and as such runs from an Amber Server, is hosted on a web server and runs in a browser window.

What Elektron Is

Features:

- Provides a means to create PanelHandlers, FrameHandlers and ApplicationHandlers. Creating these particular classes by hand is a time-consuming process, which Elektron simplifies.
- Provides an environment where the developer can graphically design the layout of a Panel or Frame by dragging and dropping Amber components on it. Elektron generates all the code for instantiating the designed panel or frame, while leaving intact any other code that may already be in the file.
- Provides a code editor to allow developers to directly modify the code. When code is modified that affects component layout, these modifications are then reflected in the GUI designer view. Naturally, it is possible to edit non-Amber java files and ordinary text files from within Elektron as well.
- Elektron allows you to compile your project with the command-line java compiler of your choice. The compiler options are fully configurable.
- Provides a “project” view of your files, so that they can be grouped and compiled together.

Because Elektron is an Amber application, the windows you see in the web browser only act as a client-side “display” for the actual Elektron program, which runs on the

Amber Server. Whenever you develop under Elektron, the files you are loading and saving are all located on the machine where the host Amber Server is running. This may not necessarily be your local machine.

What Elektron Is Not

Elektron is not the solution to all the world's problems. As mentioned before, it is only intended to be an aid to Amber development, not a fully functional IDE.

You will find Elektron very useful in developing the Amber-specific parts of your code, such as PanelHandlers, but we suggest you use a more fully featured IDE (or text editor) for the rest of your development. It is quite common to use Elektron to build up a skeleton of your application, with all the panels in place and with components arranged, before switching to another IDE to complete the development cycle. If you find you need to move components around or add new ones, you can always load the panel or frame back into Elektron and it will parse the files without difficulty (unless you have introduced syntax errors). You can then save this modified file and continue to work on it outside Elektron.

Also, Elektron is not a multi-user development environment. If multiple developers are working on a project in Elektron, it is suggested that each user has a local copy of the Amber Server, which they develop against. The "glue" which holds the developers together should be a Source Code Control system, like Microsoft Visual SourceSafe or RCS.

System Requirements

Amber Prerequisites

The Amber Server provided with Elektron must be installed and running. The machine on which Elektron is to be used must be able to browse to the web server on the Amber Server machine.

See the Installation Guide for instructions on installing and starting the Amber Server, Amber Database and Web Server. You should verify Amber is installed and running correctly by trying out the Amber Samples provided with Amber.

Ports

If you are using a firewall, the following ports must be open between the Amber Server machine and the client machine (the machine running the web browser)

- Port 80 (HTTP) HTTP is used to load the web page that launches the Elektron system. HTTP is also used to access some other sundry files, such as images and the API help.

- Port 21384 (AATP). Amber Application Transfer Protocol (AATP) is the protocol utilised by Amber to communicate between the client and server. If this port is not open, the client will not be able to connect to the server, and Amber will fail to run.

Browser Versions

Elektron has been successfully tested on:

Windows

- Netscape Version 4.7 or later (including Mozilla)
- Internet Explorer 4.1 SR 1 or later

Macintosh

- Internet Explorer Version 4.5 or later

Amber requires a 1.1 or later Java Virtual Machine, so other browsers which provide this should run successfully.

What You Need Installed on the Server If You Want To Compile

Elektron utilises a command-line compiler. If you would like to use the compilation features of Elektron, you must have a command line java compiler (such as **javac**) installed and accessible to the Amber Server. The Amber development team recommends **javac 1.2** or later.

Before compilation can proceed, you will need to configure the project's compiler command- line and associated options in the Project Properties window.

The compile feature is optional.

Getting Started

Firing up Elektron in Your Browser

Ensure the Amber Server is running. You can do this by trying out the Amber Samples. If there is a problem, please consult the troubleshooting section in the Amber Installation Guide.

1. Point your web browser to the URL

   http://<amber server hostname>/amber/elektron/elektron.html

Where the <amber server hostname> is the hostname or IP address of the box the Amber Server is running on.

You should see your web browser loading class files, and Elektron should start. If not, please consult the troubleshooting section in the Amber Installation Guide.

Creating a New Project (The First Step)

Before you do any work in Elektron, first create a project. Use the Project | New Project menu item to do this. This creates a new empty project, temporarily named "UntitledProject.epf".

Once you have a project, it can be used as a container to group related files together. Projects can be loaded, saved, closed, browsed, the files in them may be edited, and any java files a project contains can be compiled.

If necessary, this project's properties can be configured through the Project Properties (View | Project Properties).

Adding a New File to the Project (The Second Step)

There are two ways of getting a file into a project.

When you are first starting out, you will want to create a new file.

Choose the New button from the toolbar or Project window, or the File | New menu item. This brings up a dialog box offering you a range of file types you can create in Elektron. Choose one, and press OK. You will generally want to create a PanelHandler or a FrameHandler. For descriptions of all the different file types, see the Amber Development Guide.

If you already have a file you wish to add, you may choose the Add File button from the Project Manager window or the Project | Add a File to this Project, menu item. This will allow you to browse to a file you have previously saved and add it to the project.

Remember that all files Elektron allows you to browse to are on the server machine, not the client.

Once you have created or added a file, it will be able to be viewed in a separate window, ready for editing.

The Process of Creating an Amber Application in Elektron

Basic Project & Directory Management

Where Your Source Lives - The usr/src and usr/classes Directories

On the server, the installation will have created a /usr directory under the Amber Server root. All your Elektron files will appear under here, and this directory is yours to do with as you please.

The normal convention is that usr/src be used for source code, and usr/classes be used for compiled classes. These are the defaults in the Project Properties, but you may create any directories you like under the /usr directory and use a different structure if you wish.

When browsing for a file within Elektron, the system will not allow you any higher than the /usr directory. This is to limit the possibility of accidental damage to the other files on the server.

Loading, Saving & Proper Directory Management on the Server

When loading and saving files, Elektron does not attempt to automatically create a directory structure based on the java package. For instance, when saving a java class that lives in the package

```
com.mycompany.mypackage
```

You must manually create the directories com, com/mycompany and com/mycompany/mypackage before saving the file in the right place.

The file selection dialog box provides a function to allow you to easily create directories.

Remember that even if your operating system doesn't care about case sensitivity with regard to directory names, java **does** care, and you must create the directories with the proper case.

Creating a Simple Application

This section steps through the process of creating a basic Amber application in Elektron.

This tutorial will demonstrate all the basic components of the Elektron system.

The example application presented here will ask for your name, and display it back to you with a message. It will be a fully-fledged Amber application, running in a Web Browser.

The Order in Which to Do Things

For the purposes of this tutorial, we will create a panel first, lay out it's components, and then perform the supporting tasks that make it a real, running application – that is, creating the ApplicationHandler and HTML file.

However, Elektron does not force a particular workflow upon you. If you are comfortable with ApplicationHandlers and their HTML files, it is perfectly fine to do things in a different order than the one presented in this tutorial.

For the purpose of this tutorial, the steps we will follow are:

2. Create an empty Project
3. Create a panel or frame within the project
4. Lay out components on the Panel or Frame
5. Create an ApplicationHandler
6. Add the new panel to the ApplicationHandler
7. Create the HTML file to invoke the ApplicationHandler from a browser
8. Compiling your Code from within Elektron
9. Configure the Amber Server to know about the application
10. Test the application in a browser

1. Create an Empty Project

We assume you are starting this with a new, empty project. See the Getting Started section for information on how to create a new project.

2. Creating a Panel

Creating the PanelHandler

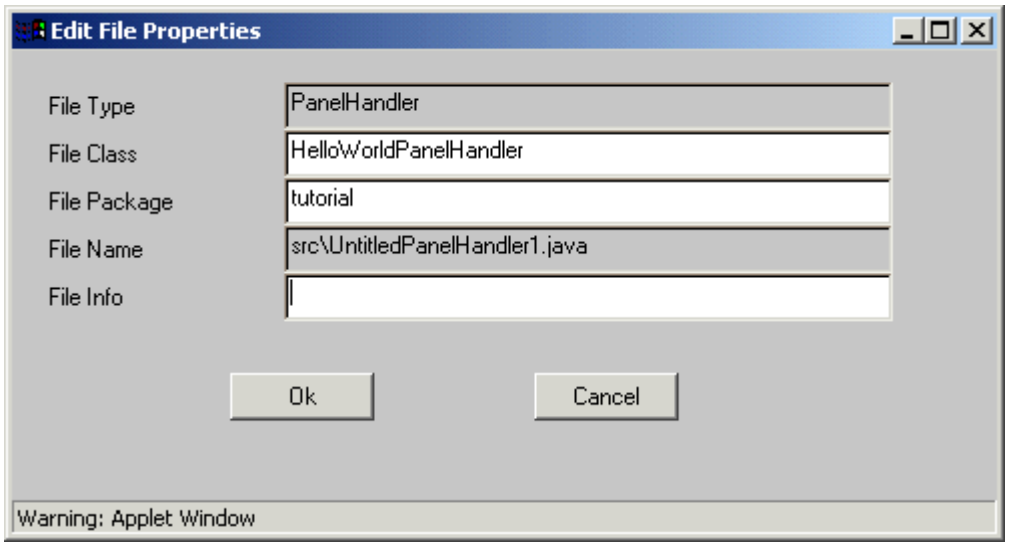
The PanelHandler is a container that holds the Amber visual components. A more detailed description of it's role is contained in the Amber Programmer's Guide.

Choose File | New from the menu, or the New button from the Project Manager window. Select new PanelHandler from the dialog that appears.

Setting the Initial Properties with the File Properties Window

Select the file in the Project Manager window, and click on the properties window. The File Properties dialog will appear. Use this dialog to change the class name and package of the file in the class definition and constructors.
If you click OK and look at the code listing for your new panel on the right hand side, you will see that the package name, class name and constructors have been updated in the code. Of course, these same changes can be made by hand, but this way is easier.

Set the File Properties as shown below.

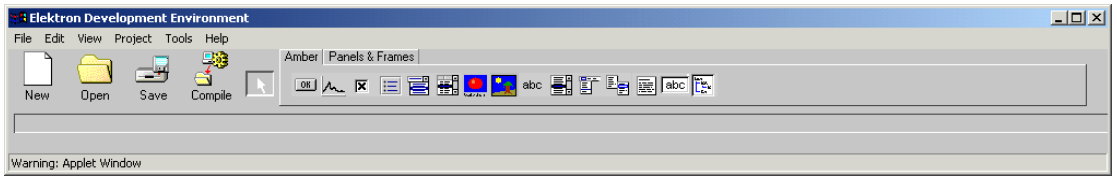


3. Lay Out Components on the Panel or Frame

Overview of the GUI Designer

Locate the code window for this PanelHandler. Click on the Design Tab. This brings up the GUI Designer view for the panel.

This GUI Designer works similarly to other GUI layout tools in most IDEs. Using it is a case of selecting the component you want on the toolbar, and clicking on the GUI designer at the location you want to add it. Once components have been dropped on the GUI designer panel, they can be dragged, resized and deleted.



The toolbar, showing some of the components it is possible to add to the GUI designer

The GUI designer understands the following hotkeys

Del	Delete selected components
Ctrl-A	Select all components
Ctrl-Left	Move the selected component(s) to the left by the number of pixels specified in the Grid Size. If snap to grid is off, moves one pixel.
Ctrl-Right	Works similarly to Ctrl-left, except moves selected components right
Ctrl-Up	Works similarly to Ctrl-left, except

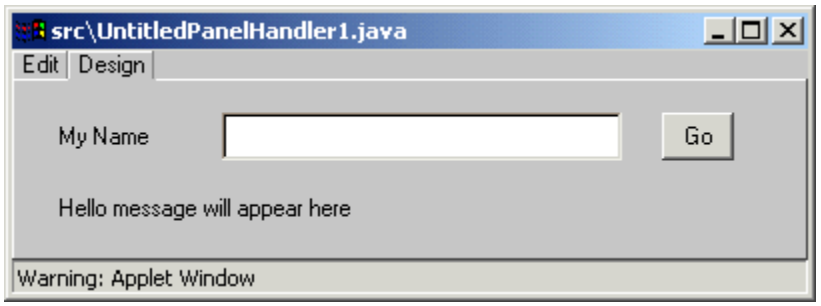
	moves selected components up
Ctrl-Down	Works similarly to Ctrl-left, except moves selected components down
L	Align selected components left (same as Edit | Align Left)
R	Align selected components right (Edit | Align Right)
T	Align selected components top (Edit | Align Top)
B	Align selected components bottom (Edit | Align Bottom)

There are several useful alignment and sizing options that can be found under the Edit menu. Likewise, options to set the grid size may be found here.

The properties of a selected component can be modified using the Component Properties Editor, which appears to the left of the GUI designer. If you flip back to the Code view, you will see that the appropriate code has been automatically generated to add components, set their position and set their properties.

Giving the File a Visual Layout in the GUI Designer

Design the following screen, by clicking and dropping new components onto the GUI designer and then changing their appropriate properties:



The components on this screen are as follows:

Name	**Type**	**Set Properties to…**
lblMyName	LabelHandler	Text property to My Name
txtMyName	TextFieldHandler	Text property to an empty string
btnGo	ButtonHandler	Caption to “Go”
lblHelloMessage	LabelHandler	Text to “Hello message will appear here”

Remember, to bring up the component properties window, if it’s not currently visible, is to use View | Component Properties.

Editing the Code by Hand

Use the GUI Designer tab to flip back to the Code view. You may alter any of the code generated while you were in edit mode, and when you return back to GUI design mode, the changes you have made will take effect.

Using the code view, you may add any functionality you like.

For the purposes of this tutorial, we will enable the Go button, and cause it to update the "Hello Message" label with a friendly message.

To enable the button:

1. Add the line
   ```
   import java.awt.event.* ;
   ```

2. Alter the line

   ```
   public class HelloWorldPanelHandler extends
   BasePanel
   ```

   to read

   ```
   public class HelloWorldPanelHandler extends
   BasePanel implements ActionListener
   ```

3. Inside the defineComponents() function, near the bottom, add the line

   ```
   btnGo.addActionListener(this) ;
   ```

4. Add the function

   ```
   public void actionPerformed (ActionEvent e)
   {
       try
       {
           lblHelloMessage.setText("Hello, " +
                           txtMyName.getText() +
                           "Welcome to Elektron!") ;
       }
       catch (ComponentException ex)
       {
           // problem occurred
           ex.printStackTrace() ;
       }
   }
   ```

And now we have a live button. Of course, we are not checking for a null return from the text box, or checking that the actionPerformed call came from the right source, but this sort of functionality is implemented in the standard java fashion, so can be added if you like.

Saving the File

Choose File | Save. Under the usr/src directory, you must create a directory corresponding to your package name (tutorial), and save the file in there. As per standard java conventions, the filename must be the same as the class name.

4. Create an ApplicationHandler

Tying Panels and Frames together into an Application

Select the File | New menu item or New on the Project Manager window. Select ApplicationHandler from the dialog that appears.

You will be presented with a code and GUI designer window similar to the one you used to lay out HelloWorldPanelHandler.java

An ApplicationHandler represents the overall Amber application. ApplicationHandlers are, if you like, the entry point of an Amber application. In effect, they are principally code modules to contain PanelHandlers and FrameHandlers. Detailed information about ApplicationHandlers may be found in the Amber Development Guide.

Using the File Properties dialog box, make the following alterations:

Package	tutorial
ClassName	HelloWorldApplicationHandler

5. Add the New Panels to the ApplicationHandler

1. In GUI Designer mode, drop a GenericPanel (from the Panels & Frames tab) onto the design area.
2. Size the object to be large enough to hold the entire panel.
3. Flip back to code view.
4. Alter all occurrences of GenericPanel in the source to HelloWorldPanelHandler
5. Save the file.

6. Create the HTML to Run Your Application from Within a Web Browser

Select the File | New menu item or New on the Project Manager window. Select HTML file from the dialog that appears.

You will be presented with a code window similar to the one you used for the panel and application handlers, but there will be no Design tab, as the HTML is treated as a plain text file.
The HTML file is the file that the web browser will load when you wish to run your Amber application. It will invoke a special Amber applet that will communicate with

the Amber Server and hence start off the process that will instantiate the application you have just created above.

An example HTML file will look like this:

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
<TITLE>Untitled Elektron Project</TITLE>
</HEAD>
<BODY>
<H1>Welcome to the Untitled Elektron Project</H1>
<APPLET
  CODE     = "amber.client.RPanel.class"
  NAME     = "UntitledComponent"
  WIDTH    = 100
  HEIGHT   = 100
  HSPACE   = 0
  VSPACE   = 0
  ALIGN    = middle
>
<PARAM NAME = "ID" VALUE = "0">
<PARAM NAME = "SERVER" VALUE = "127.0.0.1">
<PARAM NAME = "PAGEID" VALUE = "put your numeric ID in here">
</APPLET>
<HR>
</BODY>
</HTML>
```

The things you need to change are:

1. Change the `<TITLE>` and `<H1>` text to `Hello World`
2. Change the SERVER PARAM to the name or IP address of the server you are running your web server on
3. Change the PAGEID PARAM to the numeric ID that you have chosen to identify this Amber Application. The Amber Server will lookup this ID in it's database when it is asked to instantiate your application

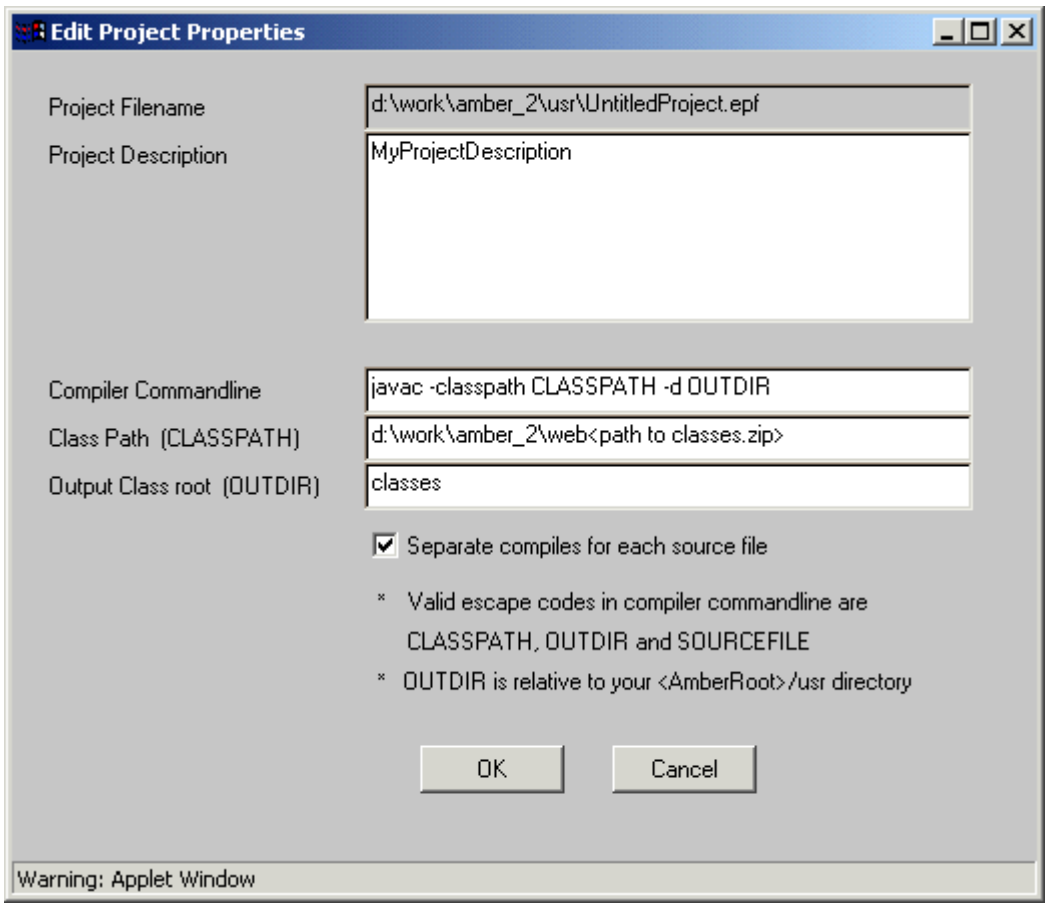
Refer to the Amber Development Guide for more detail.

7. Compiling your Code from Within Elektron

Configuring Your Project Before You Compile

Compilation in Elektron works through an external compiler, typically Sun's **javac**.

The command to execute to run the command line compiler can be configured through the Project Properties dialog.



The option "Separate compiles for each source file" is a concession to Operating Systems that have a limit on the number of characters in a command line (such as Windows 95). With this option enabled, each java file in the project will be compiled in a separate invocation of the compiler. If this option is disabled, the compiler will be called once, with all the project's files on the one command line. If you are using an OS that supports long command lines (like Linux or NT), clear this checkbox to increase efficiency.

Parameters Allowable in the Compiler Command Line Options

All these parameters are case sensitive.

OUTDIR	Substitutes the Output Directory (set in Project Properties window) into the command line
CLASSPATH	Substitutes the Classpath Directory (set in Project Properties window) into the command line. Note that you will need to include the standard java class files and your own Amber class files in this path
SOURCEFILE	Substitutes the fully qualified name of the java source file. This is optional – if

	there is no SOURCEFILE specified, it is appended to the end of the command line.

Compiling a Single File

Select the file you wish to compile in the Project Manager dialog by clicking on it and making sure it is highlighted. Click the compile button on the Toolbar, or choose the Project | Compile File menu item.

Output from the compiler will appear in the Messages Window, which will pop up after a few seconds, when the compile has completed.

Compiling All Files in the Project

Select Project | Compile All Files menu item.

8. Configure the Amber Server to Know About the Application

1. Add the new Amber application to the Amber database.
2. Restart the Amber Server.

Please refer to the Amber Development guide for instructions on how to do this.

9. Test the Application in a Browser

1. Ensure the HTML file you have generated is accessible by the web server. You may need to copy the HTML file into your web root.
2. Browse to the HTML file with your browser
3. If all is OK, your application should kick into life.

Appendices

The Parse & Generate Process

Do I Really Need to Know This Stuff?

In an ideal world, no. However, you may find that you try to open an Amber file in Elektron, and the file loads but doesn't display the visual components that you expected it to. This will be due to the way you've structured the file, so an understanding of what Elektron expects will help you track down the problem.

How Parsing Works

When Elektron loads an Amber file, it will look for the following elements:

1. Package name
   Can be read and set in the file properties
2. Class name
   Can be read and set in the file properties
3. Class that this class extends
   Used to determine the type of Amber file, one of PanelHandler, FrameHandler, ModalFrameHandler or ApplicationHandler
4. Amber components declared as class instance variables
   An instance variable must be of a class that has been loaded as a component in the Elektron toolbar for it to be recognised as a valid Amber component that Elektron can display. These declarations should be of the form:

   ```
   private ButtonHandler btnOk =
                  new ButtonHandler (getParentApplication ()) ;
   ```

5. Amber components added to the class in the defineComponents method
   Any instance variable that is added to the class inside the defineComponents method will be a candidate for Elektron displaying as a visual component. The add statement should be of the form:

   ```
   add (btnOk, new XYConstraints (109, 162, 72, 24));
   ```

6. Amber component properties set in the defineComponents method
   Any Amber components that have simple properties set will be recognised by Elektron. A simple property is defined as having a single parameters which can be an integer, a string or a Boolean. For example:

   ```
   btnOk.setLabel ("Ok");
   ```

   Any more complex properties will be ignored
7. Any other statements or comments in the file will be retained by Elektron, but ignored

If a java file is loaded, and is not recognised as a valid Amber class, it will be treated as a plain text file. Hence, it can be edited normally, but visual Amber components cannot be manipulated.

How Generating Works

When an Amber file is loaded into Elektron, a list of the Amber components and their properties is built up and stored internally. Elektron actually maintains three separate lists:

1. The Amber component declarations
2. The Amber component adds (in defineComponents method)
3. The Amber component properties set (in defineComponents method)

Whenever an Amber component is added, moved, deleted or a property changed, the visual representation of the component is changed, and also the appropriate list in memory is altered and the line (or lines) of source code are regenerated and redisplayed. Lines of code that are regenerated will replace the old lines of code, so that when the file is saved it will reflect the changes correctly.

Configuring Elektron

Adding Components

Elektron as a system is reasonably open. It is possible to add an Amber component you have written to Elektron simply by creating a text file for the new component and placing it in the appropriate directory.

The Component Properties Files & What They Do.

Each component that appears in the Elektron toolbar has a corresponding properties file in the Elektron configuration directory. A properties file is simply a text file with the ".properties" extension. Each properties file contains information about:

- The component's name and location on the toolbar
- Which java classes implement the new component
- Which properties are supported by the Component Properties Editor window in GUI designer view

Writing the New Component

Details on writing and testing new Amber components may be found in the Amber Development Guide.

When a new component has been written and is ready to be added to Elektron, you must ensure the component's class files are on the Amber Server machine, in a directory that is part of the Amber Server's classpath.

Creating the .properties File

The properties file must be saved in the
<Amber Server Root>/config/elektron/component directory.

The convention for the name of the properties file is:

```
<package name>.<classname>.properties
```

for example

```
amber.server.component.ButtonHandler.properties
```

The following is an example of a .properties file:

```
name=Button
className=amber.server.component.ButtonHandler
image=button.gif
category=Amber
defaultCaption=label

property.name=String
property.x=int
property.y=int
property.width=int,75
property.height=int,24
property.label=String
```

The block of text at the top of the file tells Elektron what it needs to know about the component in order to correctly display it on it's toolbar and instantiate it in the GUI designer.

The recognised parameters are:

name	The "plain English"name of the component. This is the name which is used when automatically generating identifiers for new components, and it also appears when you mouse-over a component button on the toolbar.
className	The name of the server-side java class that implements the component. For information on writing these classes, please see the Amber Development Guide.
image	The .gif file that appears on the component's button in the toolbar palette. The .gif file should be 26x24 pixels, and saved in the <web server root>/amber/elektron/images directory. Specifying an image that does not exist will result in a blank button on the toolbar palette.
category	The toolbar palette is arranged into tabs. The category property tells Elektron which tab this new component is to appear upon. If you specify the name of a tab that does not exist, a new tab will be created.
defaultCaption	When creating a component in the GUI designer, it may be appropriate for some components to default one of their properties to be the same as their name. For instance, labels are created with their label property set to be the same as their name, the end result being that any new label created in

	the GUI designer has appears on-screen as “label1”, “label2” etc. If you wish to use this facility for your new components, set the defaultCaption to be the name of the property you wish to default to the component’s name. This property is optional
appliesToPanels	Set this property to **true** if you wish to allow an Elektron user to instantiate your new component on a Panel or Frame Handler. The property is optional. By default, all components are allowed to be instantiated on a Panel or Frame Handler.
appliesToApplications	Set this property to **true** if you wish to allow an Elektron user to instantiate your new component on an Application Handler. The property is optional. By default, a component **is not** allowed to be instantiated on an ApplicationHandler.

The remainder of the .properties file describes the properties of the component that are recognised by the Component Properties Editor in the GUI Designer view.

These descriptors are of the form

```
property.<propertyname>=<datatype>[,<defaultvalue>]
```

For example, adding the line:

```
property.label=String
```

tells Elektron that there is a property called “label” that is to appear in the Component Properties Editor window. The label property takes a String.

Whereas the line:

```
property.width=int,75
```

tells Elektron that there is a property called “width” that is to appear in the Component Properties Editor window. The width property is set with an int, and has a default value of 75.

You should always include property descriptors for the name, x, y, width and height properties.

In order for the Component Properties Editor window to work successfully with the properties you specify, your implementation of the component must meet the following conditions

1. The datatype must be one of: int, String or boolean
2. You must implement a public method in the java code called getX() and one called setX(), where X is the name of the property you specify in the properties file. The setX() must take one parameter with a datatype matching the one in the properties file, and the getX() must take no parameters, but return a value of the specified datatype.

For instance, for the line:

```
property.caption=String
```

Your java must implement two calls:

```
public void setCaption(String newCaption)
public String getCaption()
```

Modifying the Template Files

Where They Are and What They Do

Whenever you use Elektron to generate a new file, Elektron loads a file from the disk of the server machine, and uses that file as the basis for your work.

There are 6 template files, found in the directory

```
<AmberServerRoot>/config/elektron/template
```

You may alter the templates to your own liking. The templates that ship with Elektron are very generic, and have only very simple javadoc, and don't implement some of the interfaces you might ordinarily need.

If you modify the template, you must not change the name of the file. This means that it is only currently possible to have one template of each type.

What Not To Do

Templates must contain well-formed, parsable java code.

Do not remove any of the methods found in the default templates, as Amber requires them.

The Menus

File

New…

Create a new file. This option will prompt you for the file type.

Open…

Open an existing file from the Amber Server's directory system.

Save

Save the current file to the Amber Server's directory system.

Save As…

Save the current file to the Amber Server's directory system with a new name.

Exit

Close Elektron

Edit

Find…

Look for text in the current code window. You may specify search text and options for the search in the Find dialog that appears. Searches always start from the current cursor position.

Replace…

Look for text in the current code window, and replace it with specified text. You may specify search text, replace text and options for the search in the Replace dialog that appears. Searches always start from the current cursor position.

Find or Replace Next

Repeats the last find or replace operation you performed on this code window. The search always starts from the current cursor position.

Align Left

When multiple components are selected, moves all selected components so that their left edge aligns with the leftmost component's edge.

Align Right

When multiple components are selected, moves all selected components so that their right edge aligns with the rightmost component's edge.

Center Horizontally

Centers all selected components between the leftmost and rightmost edges of the selected components.

Space Equally Horizontally

Makes sure the horizontal spacing between all components is the same. Will not move the leftmost and rightmost components.

Align Top

When multiple components are selected, moves all selected components so that their top edge aligns with the topmost component's edge.

Align Bottom

When multiple components are selected, moves all selected components so that their bottom edge aligns with the bottommost component's edge.

Center Vertically

Centers all selected components between the topmost and bottommost edges of the selected components.

Space Equally Vertically

Makes sure the vertical spacing between all components is the same. Will not move the topmost and bottommost components.

Grow to Largest Horizontally

Makes all components the same width as the widest component.

Shrink to Smallest Horizontally

Makes all components the same width as the narrowest component.

Grow to Largest Vertically

Makes all components the same height as the highest component.

Shrink to Smallest Vertically

Makes all components the same height as the shortest component.

Snap To Grid

Turns the snap to grid behaviour on and off. When it is on, all sizing and movement operations on components will be constrained so that they are a multiple of the grid size.

Grid Size

Sets the grid size, in pixels. Options are 2, 4, 8 pixels.

Delete Selected

Removes the selected components from the GUI Designer and underlying code view.

View

Project Manager

Show the project manager window. The project manager allows you to add files to a project, remove files from a project, create new files and change a file's properties.

Project Properties

Show the project properties window. Allows you to set up Project-wide parameters, such as compiler command line and project description.

File Properties

Show the file properties window. Allows you to easily change the package and class name of a file.

Component Properties

Show the file properties window while in GUI Designer view. Allows you to modify the individual properties of components.

Messages

Shows the message window. The message window displays output from the parse and compile processes.

Project

New Project

Create a new project with the name UntitledProject1.epr.

Open Project…

Open an existing project stored on the Amber Server's file system.

Save Project…

Save the current project onto the Amber Server's file system.

Save Project As…

Save the current project (with a new name) onto the Amber Server's file system.

Close Project

Closes the current project, allowing you to open another one.

Add File to Project…

Allows you to choose a file from the Amber Server's file system for inclusion in the project.

Remove File From Project

Remove the selected file from the project.

Edit File In Project

Open the selected file in a code editor window.

Show Properties of File In Project

Open the File Properties dialog. The same as View | File Properties.

Help

Contents

Opens a new browser window, which will display Elektron- and Amber-specific help.

Java Version

Runs the **java –version** command to establish the version of java you are currently running.

About

Displays the Elektron about box. The about box contains the version number of the software.